**Amendments to the Specification**

Please replace the paragraph [0004] on page 1, beginning line 17, with the following amended paragraph:

--Testing, according to the Free On-Line Dictionary of Computing (FOLDOC), ~~which can be found at http://foldoc.doc.ic.ac.uk/foldoc and which is incorporated herein by reference,~~ is defined as "[t]he process of exercising a product to identify differences between expected and actual behaviour." Many different testing techniques and types are known in the art, including black-box testing, white-box testing, unit testing, system testing, and acceptance testing. In all testing, the software under test (SUT) executes under a variety of test conditions drawn from a test suite, often with the aid of simulators, until sufficient testing has been performed. Factors such as time constraints, cost constraints, and fault tolerance play a role in determining what constitutes sufficient testing. One common metric of testing thoroughness is coverage, which tracks completeness of a set of tests, with regard to ensuring that as many areas as possible of the SUT are tested.--

Please replace the paragraph [0008] on page 3, beginning line 34, with the following amended paragraph:

--As distinct from the testing exemplified by process 19, formal verification does not execute tests against

software under test. The National Institute of Standards and Technology, an agency of the U.S. Commerce Department's Technology Administration in Gaithersburg, Md., defines formal verification in its Dictionary of Algorithms, Data Structures, and Problems ~~which can be found at http://www.nist.gov/dads and which is incorporated herein by reference,~~ as "[e]stablishing properties of hardware or software designs using logic, rather than (just) testing or informal arguments. This involves formal specification of the requirement, formal modeling of the implementation, and precise rules of inference to prove, say, that the implementation satisfies the specification." Different methods for formal verification are known in the art, including theorem proving and model checking. In the context of the present patent application and in the claims, formal verification refers to methods using model checking. In contrast to testing, formal verification operates on a model of the software and uses a model checker program to prove precisely-formulated rules. The rules are typically expressed in terms of temporal logic, describing a notation for expressing when statements are true.--